IN THE SPECIFICATION:

On page 15, second paragraph, line 15:

At the start of time slice 2, print agent 314 issues a read command to be put on notice of any request to print any file of any type. At operation 326, print agent 316 also places a read for requesting notification of any file that is requested to be printed of any type. At the end of the second time slice, at operation 328, print agent 316 receives notice of the request to print file foo of type postscript by virtue of the print request issued by application 312. At operation 330, print agent 314 also receives notification of the print request to print the file foo of type postscript. At the start of time slice 3, print agent 314, being the more efficient print agent, issues an out command indicating that it is committing itself to print the file foo. At operation 332, notice that print agent 314 is received by application 312 that ~~if~~ it will commit itself to printing file foo. At operation 336, by virtue of the in request 310, the out command is discarded to trash 318.

On page 15, fourth paragraph, lines 28 and 29:

One practical effect of the synchronized tuple space can be illustrated when we consider a case when tuple space operates with time slice .5 second, (not shown). ~~We assume that response all the times remain~~ Assume that all

response times remain the same as for Figure 9 (that is, agent 2 takes 6 second to respond). If Agent 1 responds within 1 second, its response arrives on the tuple space during time-slice 6. The response is sent to the application. The response from Agent 2 would arrive at time-slice 13. But by then there is no request for the job on the tuple space having expired and been discarded. It means that the Application never considers the commitment from Agent 2 and Agent 2 has no chance to get the contract for the print job. We can see that reduction of duration of the time slice excludes slow agents. It would favor faster agents and discriminate against slower ones, but gets the job printed quicker.